

Neural Architecture Search: Two Constant Shared Weights Initialisations

Ekaterina Gracheva^{1*}

¹*Center for Basic Research on Materials, National Institute for Materials Science, 1-1 Namiki, Tsukuba, 3050044, Ibaraki, Japan.

Corresponding author(s). E-mail(s): gracheva.ekaterina@nims.go.jp;

Abstract

In the last decade, zero-cost metrics have gained prominence in neural architecture search (NAS) due to their ability to evaluate architectures without training. These metrics are significantly faster and less computationally expensive than traditional NAS methods and provide insights into neural architectures' internal workings. This paper introduces `epsinas`, a novel zero-cost NAS metric that assesses architecture potential using two constant shared weight initialisations and the statistics of their outputs. We show that the dispersion of raw outputs, normalised by their average magnitude, strongly correlates with trained accuracy. This effect holds across image classification and language tasks on NAS-Bench-101, NAS-Bench-201, and NAS-Bench-NLP. Our method requires no data labels, operates on a single minibatch, and eliminates the need for gradient computation, making it independent of training hyperparameters, loss metrics, and human annotations. It evaluates a network in a fraction of a GPU second and integrates seamlessly into existing NAS frameworks. The code supporting this study can be found on GitHub at <https://github.com/egracheva/epsinas>.

Keywords: machine learning, neural architecture search, zero-cost NAS, efficient NAS, NAS benchmarks

1 Introduction

The field of neural architecture search (NAS) emerged roughly a decade ago as a solution to automate the process of neural geometry optimisation. In its early stages, NAS predominantly relied on evaluating candidate architectures through computationally intensive training processes, using techniques such as reinforcement learning

(Williams 1992), evolutionary algorithms (Real et al 2019; Liu et al 2021), and Bayesian optimisation (Falkner et al 2018; White et al 2021).

One-shot algorithms speed up the search by sharing weights instead of training each architecture separately. These methods include efficient reinforcement learning (Pham et al 2018), random search with parameter sharing (Li and Talwalkar 2020), and differentiable approaches (Liu et al 2018; Chu et al 2020; Xiang et al 2023). However, they require training a large hypernetwork, necessitating intricate hyperparameter tuning. While these techniques are efficient, they often fail to deliver consistent results (Dong and Yang 2019). For example, even advanced approaches like DARTS- (Chu et al 2020) exhibit significant variability compared to evolutionary or reinforcement learning methods.

Some methods attempt to estimate network performance without training on the dataset of interest by leveraging an auxiliary predictive machine learning (ML) model built on a dataset of pre-evaluated architectures (Istrate et al 2019; Deng et al 2017). Although these methods accelerate the NAS process for image recognition, they still rely on training and are not applicable to other domains of ML.

Evaluating architectures through training has several disadvantages. The most obvious is computational expense, which makes large-scale evaluations infeasible for massive datasets. As a result, architectures are typically trained with a single random seed and a fixed set of hyperparameters, raising concerns about statistical reliability. The selected architecture may perform well only within the constraints of these hyperparameters, leading to potentially suboptimal choices. Training also implies using hand-labelled data, which introduces human error — for instance, the ImageNet dataset is known to have a label error of about 6 % (Northcutt et al 2021). Furthermore, most NAS methods provide limited insights into why a particular architecture is selected, leaving gaps in understanding the underlying principles.

2 Related work

To simplify architecture search, zero-cost NAS methods identify optimal architectures without requiring full training. These methods leverage zero-cost proxies (Gaier and Ha 2019; Mellor et al 2020; Fan et al 2023; Gracheva 2021; Li et al 2023), typically demanding computational resources equivalent to just one or a few training epochs. Consequently, zero-cost NAS is two to three orders of magnitude faster than traditional NAS approaches. Below, we summarize the most prominent zero-shot NAS methods. For a comprehensive field review, see Li et al (2024).

Weight agnostic neural networks. One of the pioneering works in zero-shot NAS is presented by Gaier and Ha (2019). The authors demonstrate a method for constructing neural architectures based on the mean accuracy over several initialisations with constant shared weights and the number of parameters within the model. The resulting model achieves over 90 % accuracy on the MNIST dataset (LeCun et al 2010) when the weights are fixed to the best-performing constants. While these results are intriguing, the authors acknowledge that such architectures do not perform particularly well once trained. Moreover, in 2019, the benchmark databases of trained architectures,

now routinely used to compare NAS metrics, had not yet been released, preventing a direct comparison of this zero-shot method with more recent approaches.

ReLU activation pattern. In 2020, Mellor et al (2020) introduced the `naswot` metric, which exploits the property of the rectified linear unit (ReLU) activation function (Agarap 2018) to produce distinct activation patterns for different architectures. Specifically, every image passing through a network generates a binary activation vector, which forms a binary matrix for a mini-batch. The logarithm of the determinant of this matrix serves as a scoring metric. The authors demonstrated that higher `naswot` values are associated with better training performance, concluding that high-performing networks should be able to distinguish inputs even before training. However, this method is limited to networks using ReLU activation functions, which restricts its applicability to convolutional architectures.

A conceptually similar work by Chen et al (2021b) combines the number of linear regions in the input space with the spectrum of the neural tangent kernel (NTK) to develop the `tenas` metric. Instead of individually evaluating each network in the search space, they construct a super-network that includes all the available edges and operators, which is then pruned.

Coefficient of variance. Our previous work on fully trainless NAS (Gracheva 2021) evaluated the stability of untrained scores across random weight initialisations. Networks were initialised with multiple random seeds, and architectures were selected based on the coefficient of variance of the accuracy at initialisation, `cv`. While `cv` performance is associated with a high error rate, the study concluded that a good architecture should demonstrate stability against random weight fluctuations. Although this method theoretically applies to any type of neural architecture, it requires multiple initialisations and is relatively computationally expensive compared to `naswot` and later methods. Additionally, accuracy-based scoring metrics are limited to classification problems, and it remains unclear how to adapt the `cv` approach for regression tasks.

Inverse variance of gradients. A recent study by Li et al (2023) introduced the `zico` metric, a zero-cost proxy for neural architecture performance that, like `cv` (Gracheva 2021), is derived from network statistics. `zico` is computed as the ratio of the average gradient values to their standard deviation—in essence, a reversed `cv`, where accuracy is substituted with gradient values. Notably, `zico` exhibits a strong correlation with trained accuracy, surpassing `nparams`—the total number of parameters in a network—as a performance estimator. While `nparams` may seem like a simplistic proxy, it remained the most effective predictor of network performance until 2023.

Gradient signal-to-noise ratio. In 2023, Sun et al (2023) introduced the gradient signal-to-noise ratio metric (ξ -gsnr), which, as the name suggests, closely resembles `zico`. Their approach builds on `zico` by squaring the metric and incorporating a small constant ξ in the denominator. This seemingly minor modification significantly impacts the metric’s performance, establishing a new state-of-the-art zero-cost NAS. Additionally, the authors provide essential theoretical justification for their approach. A comparison of the `cv`, `zico`, and ξ -gsnr methodologies is presented in Table 1.

Gradient sign. The `grad_sign` metric is designed to approximate the sample-wise optimisation landscape (Zhang and Jia 2021). The authors argue that when local minima for various samples are closer, the probability increases that the corresponding

gradients will have the same sign. The number of samples yielding the same gradient sign approximates this probability, allowing for the smoothness evaluation of the optimisation landscape and the architecture’s trainability. However, this method requires labels and gradient computation.

Pruning-at-initialisation proxies. Several promising zero-cost proxies have emerged as adaptations of pruning-at-initialisation methods for NAS, as demonstrated in the work by [Abdelfattah et al \(2021\)](#): `grad_norm` ([Wang et al 2020](#)), `snip` ([Lee et al 2018](#)), and `synflow` ([Tanaka et al 2020](#)). These metrics were originally developed to assess the salience of a network’s parameters and prune potentially irrelevant synapses. They require a single forward-backwards pass to compute the loss, after which the importance of parameters is calculated as the product of the weight and gradient values. The salience is then integrated over all parameters in the network to estimate its potential performance once trained.

What makes the `synflow` metric particularly notable is its ability to evaluate architectures without referencing the data, as it computes the loss based solely on the product of all randomly initialised weights’ values. Among these proxies, the `synflow` metric demonstrates the most consistent performance across various search spaces and sets the state-of-the-art for zero-cost NAS.

Neither `naswot` nor `synflow` depend on labels, which reduces the effect of human error during data labelling. Moreover, `naswot` does not require gradient computation, which renders this method less memory-intensive.

A few other efficient zero-shot methods cannot be directly compared to our results (such as Zen-NAS by [Lin et al \(2021\)](#)). For more details on the current status of NAS, please refer to the most recent systematic review on NAS ([Salmani Pour Avval et al 2025](#)).

The results of the above studies suggest that neural networks possess an intrinsic property that determines their predictive potential even before training. This property should be independent of the specific values of trainable parameters (weights) and instead depend solely on the network’s topology. In this work, we build upon insights from existing zero-cost NAS implementations to propose a new metric that shows strong performance and outperforms current zero-cost NAS methods.

3 Epsinas metric

Two existing NAS methods inspire the metric that we share in the present work: `cv` ([Gracheva 2021](#)) and weight agnostic neural networks ([Gaier and Ha 2019](#)). Both metrics aim to reduce the role of individual weights in the network to access its topology. The `cv` metric levels out the individual weights via multiple random initialisations, while [Gaier and Ha \(2019\)](#) sets the weights to the same value across the network.

As mentioned, the `cv` metric has two principal disadvantages. While it shows a fairly consistent trend with trained accuracy, it suffers from high statistical deviations. To some degree, it must be due to the noise from random weight initialisations. With `epsinas`, we replaced random initialisations with single shared weight initialisations to improve the method’s performance.

The second weak point of `cv` is that it has been developed for classification problems and relies on accuracy, which should be maximised. On the other hand, for regression tasks, performance is typically computed as an error, which is sought to be minimised. The `cv` is a ratio of standard deviation and the mean of untrained accuracies, and the final metric correlates negatively with train accuracy. It is not apparent whether the division by mean untrained *error* would result in the same trend for `cv` metric. To address this issue, we decided to step out from evaluation metrics and consider raw outputs instead. This modification renders the method applicable to any neural architecture.

For this, we flatten the raw output matrices to obtain a single vector \mathbf{v} of length $L_{\mathbf{v}} = N_{\text{BS}} \times L_{\mathbf{O}}$ per initialisation, where N_{BS} is the batch size and $L_{\mathbf{O}}$ is the length of a single output ¹.

Before proceeding to statistics computation over initialisations, we also must normalise the output vectors: in the case of constant shared weights, outputs scale with weight values. To compare initialisations on par with each other, we use min-max normalisation:

$$\mathbf{v}'_i = \frac{\mathbf{v}_i - \min(\mathbf{v}_i)}{\max(\mathbf{v}_i) - \min(\mathbf{v}_i)}, \quad (1)$$

where i is the index for initialisations, $i = \{0, 1\}$.

We noticed that two distinct weights are sufficient to grasp the difference between initialisations. Accordingly, instead of standard deviation, we use mean absolute error between the normalised outputs of two initialisations:

$$\text{MAE} = \frac{1}{L_{\mathbf{v}}} \sum_{j=0}^{L_{\mathbf{v}}} |\mathbf{v}'_{0,j} - \mathbf{v}'_{1,j}|. \quad (2)$$

The mean is computed over the outputs of both initialisations as follows:

$$\mu = \frac{1}{L_{\mathbf{v}}} \sum_{j=0}^{L_{\mathbf{v}}} \frac{\mathbf{v}'_{1,j} + \mathbf{v}'_{2,j}}{2} \quad (3)$$

Finally, the metric is computed as the ratio of MAE and μ :

$$\varepsilon = \frac{\text{MAE}}{\mu}. \quad (4)$$

We use real data from the respective datasets to compute the statistics, as suggested in [Fan et al \(2023\)](#). In Section 6.5, we demonstrate that using real data enhances search efficiency.

We refer to our metric as `epsinas`, paying tribute to the ε symbol commonly used in mathematics to denote error bounds. The `epsinas` metric was developed independently but in parallel with `zico` and follows a similar approach. While `zico` uses the mean and standard deviation of gradients over two mini-batches, our metric combines the mean and mean absolute error of raw network outputs. This parallel development

¹This length depends on the task and architecture: for regression tasks $L_{\mathbf{O}} = 1$, for classification, $L_{\mathbf{O}}$ is equal to the number of classes, and for recurrent networks, it depends on the desired length of generated string.

Table 1 Comparison of `epsinas` to the three the most resembling zero-cost proxies. Data \mathbf{D} denotes the data used for metrics computation. N_{init} refers to the number of initialisations (for `zico` and ξ -`gsnr` we report the number of batches).

Metric	Data, \mathbf{D}	Formula	Initialisation	N_{init}	Gradients	True labels
<code>cv</code>	Accuracy	$\frac{\sigma(\mathbf{D})}{\mu(\mathbf{D})}$	Kaiming normal	100	No	Yes
<code>zico</code>	Gradients	$\frac{\mu(\mathbf{D})}{\sigma(\mathbf{D})}$	Kaiming normal	2	Yes	Yes
ξ - <code>gsnr</code>	Gradients	$\frac{\mu(\mathbf{D})^2}{\sigma(\mathbf{D})^2 + \xi}$	Kaiming normal	8	Yes	Yes
<code>epsinas</code>	Normalised outputs	$\frac{\text{MAE}(\mathbf{D})}{\mu(\mathbf{D})}$	Constant shared weights	2	No	No

Algorithm 1 Algorithm for `epsinas` metric computation

```

Select a batch of data from train set
for arch in search space do
    Initialise empty output matrix
    for weight in [val1, val2] do
        Initialise arch with constant shared weight
        Forward pass the batch through arch
        Get and flatten outputs
        Minmax normalise outputs (Eq. 1)
        Append outputs to the output matrix
    end for
    Compute difference between the rows of the output matrix (Eq. 2)
    Compute mean over the output matrix (Eq. 3)
    Compute epsinas metric (Equation 4)
end for

```

highlights a shared idea — leveraging statistical properties of untrained networks to guide architecture selection.

Table 1 compares `epsinas` to the three the most resembling zero-cost proxies: `cv`, `zico`, and ξ -`gsnr`. Although the difference between the two methods may seem minor and `zico` and ξ -`gsnr` show a strong correlation with trained accuracy, `epsinas` has an advantage: it does not require backpropagation for gradients computation, making it faster to calculate. Additionally, as we demonstrate in the next section, `epsinas` exhibits an even stronger correlation with trained accuracy.

Algorithm 1 details the `epsinas` metric computation.

Proper initialization is crucial for implementing the metric effectively. In our current implementation, we preserve the initialization of biases, batch normalization, and embeddings. Specifically, bias weights are set to zero, batch normalization parameters are initialized to ones, and embeddings follow the original paper’s approach using random integers (Klyuchnikov et al 2022).

4 Experimental design

4.1 NAS benchmarks

Following standard NAS metric evaluation practices, we assess `epsinas` on three well-established NAS benchmark datasets containing fully trained neural architectures. To ensure the broad applicability of `epsinas` across different architectures and tasks, we evaluate it on both image classification and language processing tasks.

NAS-Bench-101. The first and one of the most extensive NAS benchmarks, NAS-Bench-101, consists of 423,624 fully trained convolutional neural networks (Ying et al 2019). Each architecture comprises three stacked cells, followed by max-pooling layers, with up to 7 vertices, 9 edges, and 3 possible operations per cell. All models are trained on CIFAR-10 (Krizhevsky et al 2009) for 108 epochs using fixed hyperparameters.

NAS-Bench-201. This benchmark provides 15,625 architectures with a fixed backbone: a convolutional layer followed by three stacked cells connected by a residual block (Dong and Yang 2020). Each cell is a densely connected directed acyclic graph with 4 nodes and 5 possible operations. Architectures are trained on CIFAR-10, CIFAR-100 (Krizhevsky et al 2009), and a downsampled version of ImageNet (Chrabaszcz et al 2017), all with fixed hyperparameters over 200 epochs.

NAS-Bench-NLP. Recurrent cells comprise 24 nodes, 3 hidden states and 3 input vectors at most, with 7 allowed operations. Here, we only consider models trained and evaluated on Penn Treebank (PTB) data (Marcinkiewicz 1994): 14,322 random networks with a single seed. The training spans 50 epochs and is conducted with fixed hyperparameters.

4.2 Evaluation metrics

We evaluate the performance of `epsinas` and compare it to the results for zero-cost NAS metrics reported in Abdelfattah et al (2021). We use the following evaluation scores (computed with NaN omitted):

- Spearman ρ (global): Spearman rank correlation ρ evaluated on the entire dataset.
- Spearman ρ (top-10%): Spearman rank correlation ρ for the top-10% performing architectures.
- Kendall τ (global): Kendall rank correlation coefficient τ evaluated on the entire dataset.
- Kendall τ (top-10%): Kendall rank correlation coefficient τ for the top-10% performing architectures.
- Top-10%/top-10%: fraction of top-10% performing models within the top-10% models ranked by zero-cost scoring metric (%).
- Top-64/top-10%: number of top-64 models ranked by zero-cost scoring metric within top-5% performing models.

5 Results

5.1 NAS-Bench-201

The results for overall `epsinas` performance on NAS-Bench-201 are given in Table 2 along with other zero-cost NAS metrics. The Kendall τ score is not reported in

Table 2 Zero-cost metrics performance for the NAS-Bench-201 search space with its three datasets: CIFAR-10, CIFAR-100 and ImageNet16-120. We give the original values from [Abdelfattah et al \(2021\)](#) for reference between brackets. We highlight the best-performing metrics in bold (p-values between `epsinas` and other metrics are below 0.001).

Metric	Spearman ρ		Kendall τ		Top-10%/		Top-64/	
	global	top-10%	global	top-10%	top-10%	top-5%	top-10%	top-5%
CIFAR-10								
<code>grad_sign</code>	0.77							
<code>synflow</code>	0.74	0.18		0.54	0.12	45.75 (46)	29	(44)
<code>grad_norm</code>	0.59 (0.58)	-0.36 (-0.38)		0.43	-0.21	30.26 (30)	1	(0)
<code>grasp</code>	0.51 (0.48)	-0.35 (-0.37)		0.36	-0.21	30.77 (30)	3	(0)
<code>snip</code>	0.60 (0.58)	-0.36 (-0.38)		0.44	-0.21	30.65 (31)	1	(0)
<code>fisher</code>	0.36	-0.38		0.26	-0.24	4.99 (5)	0	(0)
<code>nparams</code>	0.75			0.57				
<code>zico</code>	0.80			0.61				
$\xi\text{-gsnr}$	0.845			0.661				
<code>epsinas</code>	0.87	0.50		0.69	0.36	67.06	62	
CIFAR-100								
<code>grad_sign</code>	0.79							
<code>synflow</code>	0.76	0.42		0.57	0.29	49.71 (50)	45	(54)
<code>grad_norm</code>	0.64	-0.09		0.47	-0.05	35.00 (35)	0	(4)
<code>grasp</code>	0.55 (0.54)	-0.10 (-0.11)		0.39	-0.06	35.32 (34)	3	(4)
<code>snip</code>	0.64 (0.63)	-0.08 (-0.09)		0.47	-0.05	35.25 (36)	0	(4)
<code>fisher</code>	0.39	-0.15 (-0.16)		0.28	-0.10	4.22 (4)	0	(0)
<code>nparams</code>	0.73			0.55				
<code>zico</code>	0.81			0.61				
$\xi\text{-gsnr}$	0.840			0.658				
<code>epsinas</code>	0.89	0.63		0.72	0.45	81.33	62	
ImageNet16-120								
<code>grad_sign</code>	0.78							
<code>synflow</code>	0.75	0.55		0.56	0.39	43.57 (44)	26	(56)
<code>grad_norm</code>	0.58	0.12 (0.13)		0.43	0.09	31.29 (31)	0	(13)
<code>grasp</code>	0.55 (0.56)	0.10		0.39	0.07	31.61 (32)	2	(14)
<code>snip</code>	0.58	0.13		0.43	0.09	31.16 (31)	0	(13)
<code>fisher</code>	0.33	0.02		0.25	0.01	4.61 (5)	0	(0)
<code>nparams</code>	0.69			0.52				
<code>zico</code>	0.79			0.60				
$\xi\text{-gsnr}$	0.793			0.608				
<code>epsinas</code>	0.85	0.53		0.67	0.37	71.51	59	

[Abdelfattah et al \(2021\)](#), but it is considered more robust than Spearman ρ and is increasingly used for NAS metric evaluation. We use the data provided by [Abdelfattah et al \(2021\)](#) to evaluate their Kendall τ . Note that our results differ from the original paper in terms of evaluation scores. In such cases, we indicate the original values between brackets. In particular, there is a discrepancy in computing the values in the last column, *Top-64/top-5%*, while the rest of the results are consistent. Figure 8 in the Appendix suggests that our calculations are correct.

Table 3 Comparison of the zero-cost metrics performances against existing NAS algorithms on CIFAR-10, CIFAR-100 and ImageNet16-120 datasets. On the top, we list the best-performing methods that require training: REA (Real et al 2019), random search, REINFORCE (Williams 1992), BOHB (Falkner et al 2018), DARTS- (Chu et al 2020). We report the average best-achieved test accuracy over 500 runs, with 1,000 architectures (100 for `grad_sign`) sampled from the search space at random. Random and optimal performances are given as a baseline.

Method	Cost (h)	CIFAR-10		CIFAR-100		ImageNet16-120	
		validation	test	validation	test	validation	test
State-of-the-art							
REA	3.3	91.19 ± 0.31	93.92 ± 0.3	71.81 ± 1.12	71.84 ± 0.99	45.15 ± 0.89	45.54 ± 1.03
RS	3.3	90.93 ± 0.36	93.92 ± 0.31	70.93 ± 1.09	71.04 ± 1.07	44.45 ± 1.1	44.57 ± 1.25
REINFORCE	3.3	91.09 ± 0.37	93.92 ± 0.32	71.61 ± 1.12	71.71 ± 1.09	45.05 ± 1.02	45.24 ± 1.18
BOHB	3.3	90.82 ± 0.53	93.92 ± 0.33	70.74 ± 1.29	70.85 ± 1.28	44.26 ± 1.36	44.42 ± 1.49
DARTS-	3.2	91.03 ± 0.44	93.80 ± 0.40	71.36 ± 1.51	71.53 ± 1.51	44.87 ± 1.46	45.12 ± 0.82
Baselines (N=1000)							
Optimal	n/a	91.34 ± 0.18	94.20 ± 0.13	72.53 ± 0.53	72.84 ± 0.41	45.93 ± 0.51	46.59 ± 0.34
Random	n/a	84.11 ± 11.71	87.40 ± 11.94	61.57 ± 11.35	61.67 ± 11.35	33.97 ± 8.68	33.67 ± 8.98
Zero-cost (N=1000)							
naswot	3.3	89.69 ± 0.73	92.96 ± 0.81	69.86 ± 1.21	69.98 ± 1.22	43.95 ± 2.05	44.44 ± 2.10
synflow	n/a	89.91 ± 0.83	90.12 ± 0.78	70.35 ± 2.25	70.37 ± 2.08	41.73 ± 3.91	42.11 ± 4.02
grad_norm	n/a	88.13 ± 2.35	88.42 ± 2.28	66.35 ± 5.45	66.48 ± 5.32	33.88 ± 11.46	33.90 ± 11.74
grasp	n/a	87.85 ± 2.12	88.17 ± 2.04	65.36 ± 5.57	65.45 ± 5.48	32.23 ± 10.95	32.20 ± 11.23
snip	n/a	87.47 ± 2.19	87.81 ± 2.12	64.61 ± 5.52	64.74 ± 5.43	30.65 ± 11.32	30.55 ± 11.55
fisher	n/a	87.01 ± 2.31	87.36 ± 2.23	63.54 ± 5.69	63.67 ± 5.62	26.70 ± 10.83	29.56 ± 10.83
epsinas	0.03	91.03 ± 0.42	93.86 ± 0.43	71.76 ± 0.90	71.79 ± 0.86	45.11 ± 0.99	45.42 ± 1.21
Zero-cost (N=100)							
cv	0.05	84.89 ± 6.39	91.90 ± 2.27	63.99 ± 5.61	64.08 ± 5.63	38.68 ± 6.34	38.76 ± 6.62
grad_sign	n/a	89.84 ± 0.61	93.31 ± 0.47	70.22 ± 1.32	70.33 ± 1.28	42.07 ± 2.78	42.42 ± 2.81
epsinas	0.003	90.44 ± 0.97	93.39 ± 0.82	70.85 ± 1.30	71.00 ± 1.26	44.03 ± 2.02	44.20 ± 2.04

For NAS-Bench-201, we also report average performance when selecting one architecture from a pool of N random architectures. The statistics are reported over 500 runs. Table 3 compares `epsinas` to other zero-cost metrics.

Comparing the results for `epsinas` with other zero-cost NAS metrics, we can see that it shows surprisingly good performance given its conceptual simplicity. Importantly, it is consistently better than the naive estimator `nparams`.

Figure 1 further confirms the applicability of the method to the NAS-Bench-201 field (similar plots for other methods can be found in the Ablation section, Figure 8). However, NAS-Bench-201 is a relatively compact search space; furthermore, it has been used for `epsinas` development.

5.2 NAS-Bench-101

We use the NAS-Bench-101 search space to verify that the effectiveness of the `epsinas` metric, as demonstrated in the previous section, is not merely a result of overfitting to the NAS-Bench-201 space. It also allows us to assess its performance in a broader search space. As shown in Table 4 and Figure 2, `epsinas` performs well on NAS-Bench-101, once again surpassing the `nparams` proxy.

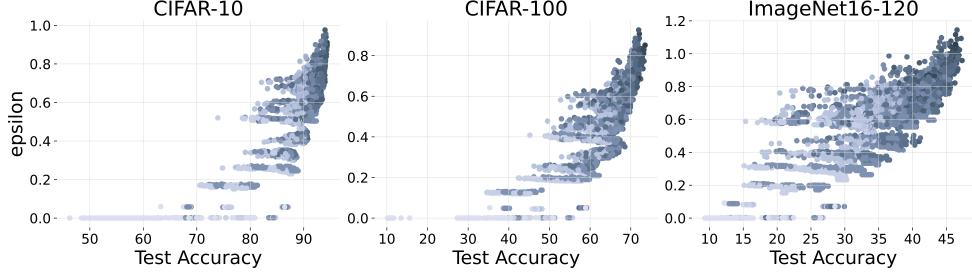


Fig. 1 Zero-cost NAS `epsinas` metric performance illustration for NAS-Bench-201 search space evaluated on CIFAR-10, CIFAR-100 and ImageNet16-120 datasets. The horizontal axis shows test accuracy upon training. Each dot corresponds to an architecture; the darker the colour, the more parameters it contains. The figure represents the search space of 15,625 networks (excluding architectures with `NaN` scores).

Table 4 Zero-cost metrics performance evaluated on NAS-Bench-101 search space, CIFAR-10 dataset. Values from Abdelfattah et al (2021) are given for reference between brackets. We highlight the best-performing metrics in bold (p-values between `epsinas` and other metrics are below 0.001).

Metric	Spearman ρ		Kendall τ		Top-10%/ Top-64/	
	global	top-10%	global	top-10%	top-10%	top-5%
grad.sign	0.45					
synflow	0.37	0.14		0.25	0.10	22.67 (23)
grad.norm	-0.20	-0.05 (0.05)		-0.14	-0.03	1.98 (2)
grasp	0.45	-0.01		0.31	-0.01	25.60 (26)
snip	-0.16	0.01 (-0.01)		-0.11	0.00	3.43 (3)
fisher	-0.26	-0.07 (0.07)		-0.18	-0.05	2.65 (3)
nparams	0.43			0.31		
zico	0.63			0.46		
ξ -gsnr	0.615			0.434		
<code>epsinas</code>	0.62	0.12		0.44	0.08	40.33 10

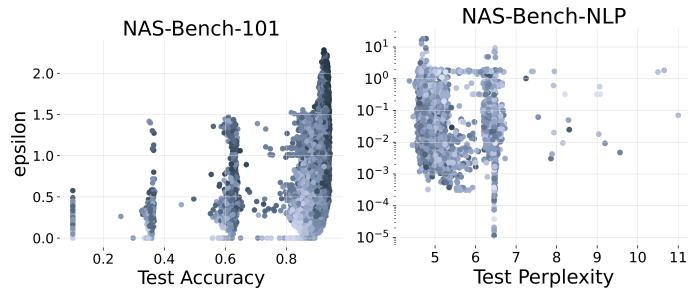


Fig. 2 Zero-cost NAS `epsinas` metric performance illustration for NAS-Bench-101 search space, CIFAR-10 dataset and NAS-Bench-NLP search space, PTB dataset. The horizontal axis shows test accuracy upon training. Each dot corresponds to an architecture; the darker the colour, the more parameters it contains. The figure shows 423,624 and 14,322 networks for NAS-Bench-101 and NAS-Bench-NLP, respectively (excluding architectures with `NaN` scores).

Table 5 Zero-cost metrics performance evaluated on NAS-Bench-NLP search space, PTB dataset. We highlight the best-performing metrics in bold (p-values between **epsinas** and other metrics are below 0.001).

Metric	Spearman ρ		Kendall τ		Top-10%/ top-10%	Top-64/ top-5%
	global	top-10%	global	top-10%	top-10%	top-5%
synflow	0.34	0.10			22	
grad_norm	-0.21	0.03			10	
grasp	0.16	0.55			4	
snip	-0.19	-0.02			10	
epsinas	-0.34	-0.12	-0.24	-0.08	26.23	13

5.3 NAS-Bench-NLP

Both NAS-Bench-201 and NAS-Bench-101 are created to facilitate NAS in image recognition. They operate convolutional networks of very similar constitutions. To truly probe the generalisability of the **epsinas** metric, we test it on NAS-Bench-NLP. Both input data format and architecture type differ from the first two search spaces.

Unfortunately, [Abdelfattah et al \(2021\)](#) provides no data for NAS-Bench-NLP, disabling us from using their results for calculations. Therefore, in Table 5, we give only values provided in the paper together with our **epsinas** metric (data for **ficher** is absent). It's important to note that, unlike accuracy, perplexity – commonly used in language-related machine learning problems – should be minimized. Thus, the signs of correlations with scoring metrics must be reversed, which differs from the results reported in [Abdelfattah et al \(2021\)](#).

5.4 Computational cost

One of the primary purposes of zero-shot proxies is to reduce the computational cost of the NAS process. Since **epsinas** only requires two forward passes through the data and does not involve gradient computation, it is efficient regarding both time and memory usage. Figure 3 presents the results of tests conducted using NAS-Bench-101. As anticipated, memory usage and processing time scale linearly with the number of parameters and FLOPs. It also demonstrates that **epsinas** can be computed using a CPU within 30 seconds for the largest NAS-Bench-101 networks consisting of several million parameters.

Comparing **epsinas** to other zero-cost metrics is not straightforward, as different methods use different search spaces, computational setups, and batch sizes. For instance, **zico** reports evaluation times using MobileNetV2 [Sandler et al \(2018\)](#) with a budget of 1,000 MFLOPs and a batch size of 128. The authors noted that evaluating 10^5 architectures required 10 GPU hours.

MobileNetV2 networks range from 1.7M to 6.9M parameters, larger than NAS-Bench-201 but smaller than NAS-Bench-101. To provide a closer comparison, we evaluated **epsinas** on NAS-Bench-101 with a batch size of 128, limiting architectures to 6.9M parameters. Under these conditions, evaluating 10^5 architectures took approximately 7 GPU hours, with the **epsinas** computation requiring only 2 GPU hours; the remaining time was spent on model loading.

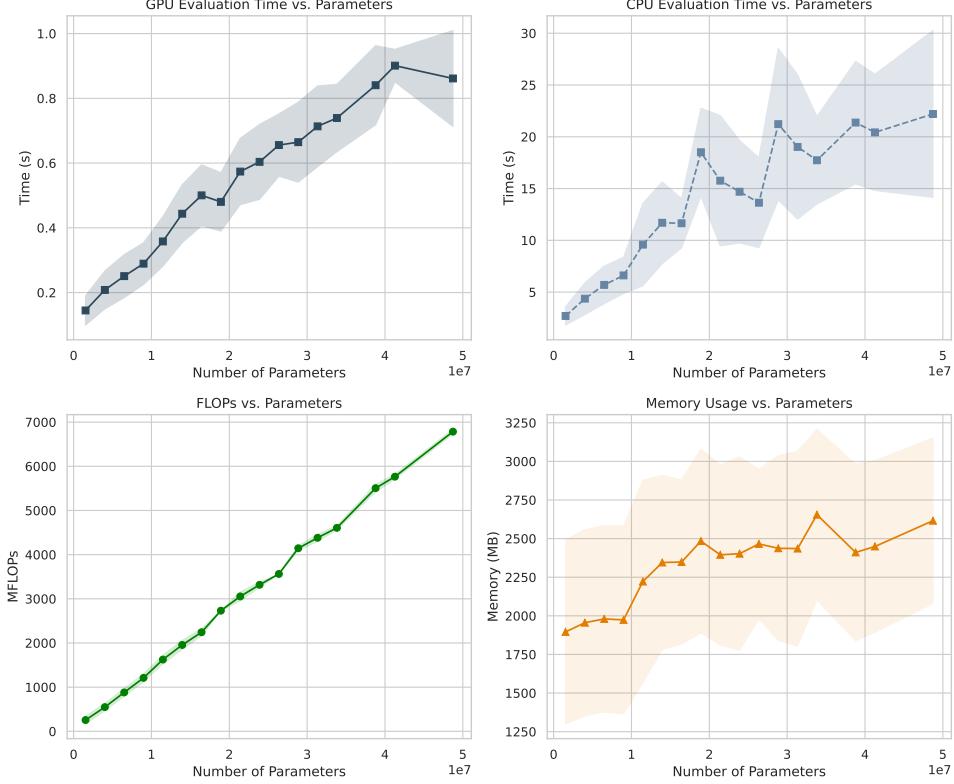


Fig. 3 Computational efficiency of the zero-cost NAS `epsinas` metric for the NAS-Bench-101 search space. The results are based on the evaluation of 10,000 networks from the NAS-Bench-101 benchmark, divided into 20 bins according to the number of parameters. The shaded area represents one standard deviation within each group. FLOPs are reported for a batch size of 1, while memory usage and processing time are evaluated with a batch size of 256.

While this is not a direct comparison, `epsinas` does not involve gradient computation, so it is reasonable to expect it to be faster per architecture evaluation. The results support this expectation.

5.5 Correlation with `nparams`

As noted in previous studies (Ning et al 2021; Li et al 2023; White et al 2022), one of the best naive evaluators of a network’s performance is the number of parameters, `nparams`. Generally, the more parameters a network has, the better we expect its predictive performance to be. However, bigger does not always mean better: large networks are susceptible to overfitting and can be prohibitively expensive to train (Dauphin and Bengio 2013; LeCun et al 1990). Moreover, research on network pruning has shown that, with the right architecture, smaller subnetworks can achieve comparable or even superior performance to their larger counterparts (Frankle and Carbin 2018). Hence, when seeking an optimal evaluation proxy, it is essential to ensure that it can effectively

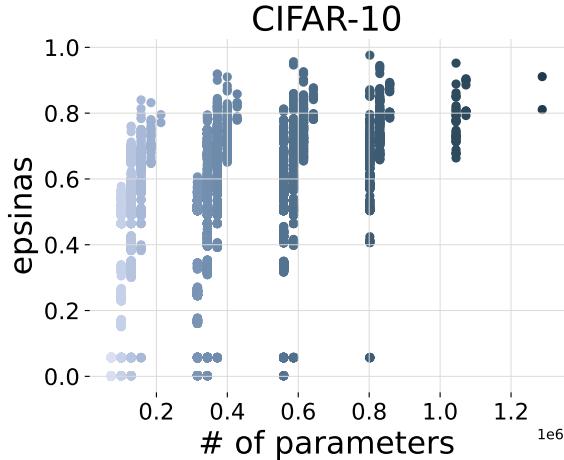


Fig. 4 Visualisation of the relationship between `epsinas` score and the number of parameters in a network, `nparams` for CIFAR-10 data of NAS-Bench-201 benchmark.

identify smaller networks. Figure 4 illustrates the relationship between `epsinas` and the number of parameters, showing no strong preference for larger network sizes.

5.6 Integration with other NAS methods

While zero-cost metrics are robust on their own, they benefit from being integrated with other NAS algorithms. Lopes et al (2022) showed that combining a zero-cost metric based on Jacobian covariance, similar to Mellor et al (2020), with REA (Real et al 2019) speeds up evolutionary search and helps find top-performing architectures. This work demonstrates how the `epsinas` metric can improve random search and ageing evolution algorithms.

This work follows the implementation of Abdelfattah et al (2021).

Random search Each new architecture is selected randomly and evaluated based on test accuracy. The final test performance is reported without using any auxiliary metrics.

Random search with warm-up A warm-up pool of 3,000 randomly sampled architectures is first ranked using `epsinas`. Next, models are "trained" in order of decreasing `epsinas` value by retrieving their test accuracies. Training stops once the budget of 300 architectures is reached.

Ageing evolution We implement the basic ageing evolution algorithm from Real et al (2019). The process starts with a randomly sampled pool of 64 architectures. Parents are selected based on test accuracy, and a child is generated through mutation (with an edit distance of 1). The child is added to the pool, while the oldest architecture is removed. This process repeats until 300 architectures have been trained.

Ageing evolution with warm-up A warm-up phase begins with 3,000 sampled architectures, which are ranked using the `epsinas` metric. The 64 architectures with the highest `epsinas` values form the initial pool. The standard ageing evolution process continues, where parents are selected based on test accuracy.

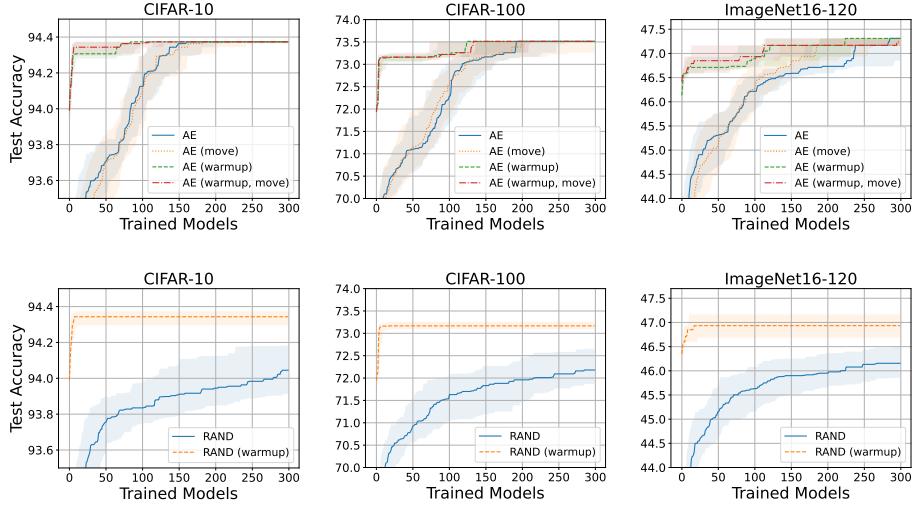


Fig. 5 `epsinas` integration within ageing evolution (top) and random search (bottom) NAS algorithms for three datasets from NAS-Bench-201 search space.

Ageing evolution with move Instead of using test accuracy for parent selection, architectures are assessed based on their `epsinas` scores.

We conduct 100 random runs for each algorithm and report the mean and standard deviation.

Figure 5 demonstrates that the `epsinas` metric significantly improves both efficiency and accuracy, with the best results achieved when combined with a warm-up. Figure 7 compares warm-up performances across various zero-cost metrics.

6 Ablation studies

While our metric is straightforward to implement, it depends on several hyper-parameters. In this section, we present and analyse the results of our ablation studies.

6.1 Weights

To compute the `epsinas` metric, we initialise the networks with two distinct constant shared weights. The question is: how do we fix their values, and how does this choice affect the whole method?

To answer this question, we ran a series of tests with multiple pairs of weights. It makes sense to set the first weight always greater than the second one in pairs. This consideration stems from the fact that `epsinas` is based on the mean absolute difference between two initialisations. Therefore, it will output the same score for $[w_1, w_2]$ and $[w_2, w_1]$ combinations and will be exactly zero in the case where the two weights are equal. Tables 6, 7, 8 summarise the results of our tests. For the NAS-Bench-201 search space, the outcomes are very similar between the datasets, and the minor differences do not affect the conclusions drawn from this data.

The results indicate that variations in initialisation do affect performance. Two key factors influence this impact. First, a more significant difference between the two weights leads to a higher correlation between `epsinas` and trained performance. It is intuitive – when the weights are too similar, the output difference becomes minimal, making it difficult to distinguish between architectures based on `epsinas`.

Secondly, we can see that too low and too high weights result in fewer architectures with non-NaN scores. Extreme weights result in all zeros or infinite output values and, consequently, in NaN value of the `epsinas` metric. Naturally, the deeper the network is, the more signal attenuation is, and the higher the probability of NaN score. We remember that before `epsinas` computation, the outputs are normalised to the same range [0, 1], which means the metric value per se does not become smaller for deeper networks. This extinction effect is due to the limited sensitivity of float type. Theoretically, with infinite float sensitivity, there would be no extinction and hence no effects on the choice of the weights.

As a rule of thumb, for every new ML problem, we suggest running `epsinas` evaluation on a subset of architectures with several weights and selecting the minimum and maximum weights that do not cause excessive NaN outputs. While we acknowledge that this procedure is subjective, importantly, it does not require any training.

Weight combinations that we selected for each search space are given in Table 9.

6.2 Initialisation algorithm

The choice to initialise the weights with constant values provides a ground for discussion. On one hand, using constant shared weights ensures a consistent initialisation across different structures, eliminating the noise introduced by random initialisations. It allows us to isolate and analyse the impact of neural geometry more effectively. Averaging results over multiple runs with different random seeds, as in Gracheva (2021), can reduce the randomness noise but increases computational time proportionally.

Table 10 presents the results of our ablation studies, where `epsinas` is computed using different initialisation schemes. The correlation coefficients are based on the `epsinas` scores from two initialisations but using standard initialisation methods instead of constant shared weights. For every initialisation scheme, we only modify

Table 6 Zero-cost metrics performance was evaluated using the NAS-Bench-101 search space and the CIFAR-10 dataset. We start with 5,000 randomly picked architectures. The number of architectures with non-NaN values is given in the second column.

Weights	Archs	Spearman ρ		Kendall τ		Top-10%/ Top-64/	
		global	top-10%	global	top-10%	top-10%	top-5%
0.0001, 0.001	2120	0.47	0.41	0.33	0.28	38.21	23
0.0001, 0.01	2120	0.61	0.03	0.43	0.02	41.98	24
0.0001, 0.1	2120	0.61	0.03	0.43	0.02	41.98	24
0.0001, 1	2120	0.61	0.03	0.43	0.02	41.98	24
0.001, 0.01	4968	0.29	-0.05	0.20	-0.03	17.30	10
0.001, 0.1	4968	0.29	-0.05	0.19	-0.03	17.30	8
0.001, 1	4968	0.29	-0.05	0.19	-0.03	17.30	8
0.01, 0.1	5000	0.21	-0.12	0.14	-0.08	4.60	0
0.01, 1	5000	0.21	-0.12	0.14	-0.08	4.60	0
0.1, 1	5000	-0.47	0.04	-0.32	0.03	0.20	0

Table 7 Zero-cost metrics performance was evaluated in the NAS-Bench-201 search space and the CIFAR-10 dataset. We start with 5,000 randomly picked architectures. The number of architectures with non-NaN values is given in the second column.

Weights	Archs	Spearman ρ		Kendall τ		Top-10%/ Top-64/	
		global	top-10%	global	top-10%	top-10%	top-5%
1e-07, 1e-06	2957	0.33	-0.20	0.22	-0.13	4.76	1
1e-07, 1e-05	2957	0.39	-0.06	0.29	-0.04	9.15	2
1e-07, 0.0001	2957	0.47	-0.01	0.32	0.00	9.12	2
1e-07, 0.001	2957	0.62	0.06	0.43	0.05	18.92	3
1e-07, 0.01	2957	0.83	0.69	0.64	0.49	68.40	56
1e-07, 0.1	2957	0.84	0.64	0.65	0.46	68.03	56
1e-07, 1	2957	0.87	0.59	0.70	0.43	65.88	45
1e-06, 1e-05	2957	0.41	-0.06	0.30	-0.04	9.15	2
1e-06, 0.0001	2957	0.47	-0.01	0.32	0.00	9.12	2
1e-06, 0.001	2957	0.62	0.06	0.43	0.05	18.92	3
1e-06, 0.01	2957	0.83	0.69	0.64	0.49	68.40	56
1e-06, 0.1	2957	0.84	0.64	0.65	0.46	68.03	56
1e-06, 1	2957	0.87	0.59	0.70	0.43	65.88	45
1e-05, 0.0001	2957	0.47	-0.01	0.32	0.00	9.12	2
1e-05, 0.001	2957	0.62	0.06	0.43	0.05	18.92	3
1e-05, 0.01	2957	0.83	0.69	0.64	0.49	68.40	56
1e-05, 0.1	2957	0.84	0.64	0.65	0.46	68.03	56
1e-05, 1	2957	0.87	0.59	0.70	0.43	65.88	45
0.0001, 0.001	2957	0.62	0.06	0.43	0.05	19.26	3
0.0001, 0.01	2957	0.82	0.69	0.63	0.49	68.42	56
0.0001, 0.1	2957	0.84	0.64	0.65	0.46	68.14	56
0.0001, 1	2957	0.87	0.58	0.69	0.42	65.88	46
0.001, 0.01	4802	0.25	0.41	0.17	0.28	30.35	52
0.001, 0.1	4802	0.31	0.27	0.20	0.19	26.82	25
0.001, 1	4802	0.31	0.22	0.21	0.15	26.26	6
0.01, 0.1	4907	0.15	0.42	0.10	0.29	37.47	57
0.01, 1	4907	0.12	0.39	0.06	0.27	35.85	57
0.1, 1	4907	0.01	0.26	-0.03	0.16	0.84	0

the convolutional layers. The rest of the architecture is initialised as per the original benchmark space: constant initialisation with ones for the batch normalisation, normal uniform for the linear layer, and biases set to zero.

6.3 Batch size

To test the method’s sensitivity against the batch size, we ran the scoring routine with 8 different batch sizes [8, 16, 32, 64, 128, 512, 1024], each with 10 random batches. The tests are done on the NAS-Bench-201 search space with its three datasets. We report medians with 25 and 75 percentiles over 10 runs for each batch. We evaluate **epsinas** on 5,000 architectures.

Figure 6 shows that, as expected, the larger the batch size, the better and more stable the performance of **epsinas** metric. There is almost no improvement for batches over 256. We set the batch size to this value in the experiments throughout the paper.

Table 8 Zero-cost metrics performance was evaluated using the NAS-Bench-NLP search space and the PTB dataset. We start with 5,000 randomly picked architectures. The number of architectures with non-NaN values is given in the second column.

Weights	Archs	Spearman ρ		Kendall τ		Top-10%/ top-10%	Top-64/ top-5%
		global	top-10%	global	top-10%		
1e-07, 1e-06	2393	-0.24	0.32	-0.16	0.22	0.00	0
1e-07, 1e-05	2391	-0.22	0.52	-0.15	0.39	1.67	0
1e-07, 0.0001	2388	-0.35	0.49	-0.24	0.37	0.42	1
1e-07, 0.001	2332	-0.42	0.50	-0.29	0.37	2.99	1
1e-07, 0.01	1215	-0.30	0.13	-0.21	0.09	4.20	1
1e-07, 0.1	585	-0.49	0.55	-0.36	0.44	1.69	1
1e-07, 1	403	-0.12	0.08	-0.08	-0.00	12.20	7
1e-06, 1e-05	3286	-0.11	0.49	-0.07	0.36	0.91	0
1e-06, 0.0001	3283	-0.24	0.47	-0.16	0.34	0.62	0
1e-06, 0.001	3226	-0.33	0.44	-0.23	0.33	4.64	1
1e-06, 0.01	1639	-0.26	0.13	-0.18	0.09	3.40	2
1e-06, 0.1	841	-0.41	0.13	-0.28	0.12	10.71	1
1e-06, 1	594	-0.09	0.13	-0.06	0.08	11.67	7
1e-05, 0.0001	3948	-0.32	0.30	-0.21	0.22	0.51	0
1e-05, 0.001	3892	-0.35	0.20	-0.24	0.16	3.85	2
1e-05, 0.01	1900	-0.32	0.27	-0.22	0.18	2.86	1
1e-05, 0.1	913	-0.38	0.31	-0.28	0.25	13.04	1
1e-05, 1	649	-0.18	0.36	-0.12	0.25	10.77	7
0.0001, 0.001	3987	-0.35	0.18	-0.24	0.14	3.76	0
0.0001, 0.01	1943	-0.37	0.38	-0.26	0.25	3.59	1
0.0001, 0.1	925	-0.43	0.26	-0.30	0.21	11.83	1
0.0001, 1	663	-0.26	0.35	-0.18	0.24	10.45	7
0.001, 0.01	1936	-0.32	0.40	-0.21	0.26	2.59	3
0.001, 0.1	918	-0.38	0.27	-0.30	0.22	8.70	2
0.001, 1	663	-0.23	-0.20	-0.19	-0.17	7.46	5
0.01, 0.1	910	-0.36	0.04	-0.25	0.02	8.79	2
0.01, 1	654	-0.25	0.06	-0.16	0.04	9.09	5
0.1, 1	652	-0.27	0.20	-0.19	0.13	10.61	7

Table 9 Optimal weights for epsinas evaluation for three search spaces and their datasets.

Search space	Dataset	Optimal weights
NAS-Bench-101	CIFAR-10	$[10^{-4}, 10]$
	CIFAR-10	$[10^{-7}, 1]$
NAS-Bench-201	CIFAR-100	$[10^{-7}, 1]$
	ImageNet16-120	$[10^{-7}, 1]$
NAS-Bench-NLP	PTB	$[10^{-5}, 10^{-3}]$

6.4 Embedding initialisation

To verify the effect of the embedding initialisation, we have run tests with 6 different initialisations:

- uniform positive 0.1: random uniform from ranges [0, 0.1]
- uniform positive 1: random uniform from ranges [0, 1]
- uniform centred 0.1: random uniform from ranges [-0.1, 0.1]
- uniform centred 1: random uniform from ranges [-1, 1]
- random 0.1: random normal centred at 0 with a standard deviation of 0.1

Table 10 Initialisations ablation studies. The metric is computed over the first 5,000 architectures in the NAS-Bench-201 benchmark.

Initialisation	Spearman ρ global top-10%	Kendall τ global top-10%	Top-10%/ top-10%	Top-64/ top-5%
Uniform	-0.02	-0.01	-0.01	-0.00
Normal	0.08	-0.20	0.05	-0.13
Kaiming uniform	0.07	-0.17	0.05	0.12
Kaiming normal	0.07	-0.13	0.05	-0.09
Orthogonal	0.07	-0.16	0.05	-0.10

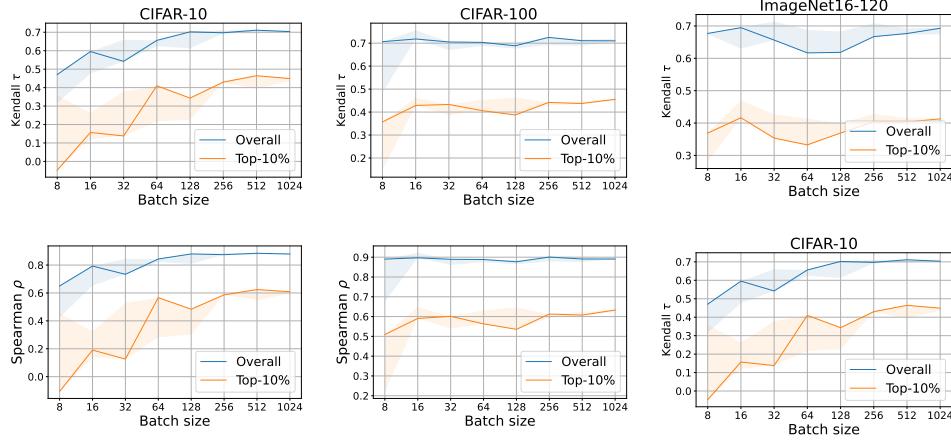


Fig. 6 Batch size ablation study for NAS-Bench-201 search space, CIFAR-10 (left), CIFAR-100 (centre) and ImageNet16-120 (right) datasets.

Table 11 Embedding ablation studies. Metric computed over 5,000 initial architectures. The number of remaining architectures is given in the second column.

Embedding	Archs	Spearman ρ global top-10%	Kendall τ global top-10%	Top-10%/ top-10%	Top-64/ top-5%
Uniform positive 0.1	782	-0.38	0.17	-0.26	0.15
Uniform positive 1	776	-0.37	0.21	-0.26	0.18
Uniform centered 0.1	783	-0.40	0.18	-0.28	0.15
Uniform centered 1	782	-0.46	0.18	-0.33	0.16
Random 0.1	783	-0.42	0.18	-0.29	0.15
Random 1	782	-0.47	0.19	-0.33	0.16

- random 1: random normal centred at 0 with a standard deviation of 1

Table 11 summarises our results of the embedding ablations. Ablation is done on the NAS-Bench-NLP search space, PTB dataset (the only search space implementing embedding).

Care should be taken when initialising the networks containing the embedding layer: if embedding is initialised with all constants, there is no difference between the

Table 12 Synthetic data tests with CIFAR-10 dataset, NAS-Bench-201 search space. Tests are based on the evaluation of 5,000 architectures.

Metric	Spearman ρ global top-10%	Kendall τ global top-10%	Top-10%/ top-10%	Top-64/ top-5%
Real data	0.87	0.59	0.70	0.43
Grey scale	0.87	0.44	0.68	0.31
Random normal	0.54	0.15	0.38	0.14
Random uniform	0.56	0.17	0.40	0.16
Random uniform (+)	0.61	0.17	0.43	0.16
			65.88	45
			59.46	34
			14.97	3
			14.19	3
			16.22	4

embedded input. In this case, our metric’s performance will be analogous to that with a batch size of one.

Our ablation studies show that it does not significantly influence the outcomes as soon as the embedding is initialised with non-constant weights.

6.5 Synthetic data

In this section, we test the importance of the input data by feeding our metric several synthetic input data on CIFAR-10. We feed networks with 5 types of data:

- actual data: a batch of CIFAR-10 data
- grey scale images: images within the batch are solid colour ranging from black to white
- random normal: images are filled with random values following a normal distribution with $[\mu, \sigma] = [0, 1]$
- random uniform: images are filled with random values following uniform distribution with $[\mu, \sigma] = [-1, 1]$
- random uniform (+): images are filled with random values following uniform distribution with $[min, max] = [0, 1]$

All the tests are performed with batch size of 256, weights $[10^{-7}, 1]$ and 5,000 architectures. Table 12 shows that even though the performance with synthetic data drops compared to actual data, it is still reasonably good.

Curiously, greyscale images, filled with constant values, show the closest results to the actual data. Note that **epsinas** metric with greyscale data outperforms **synflow**. It is an essential achievement because **synflow** does not use input data and is, therefore, data independent. Our results show that **epsinas** has the potential to be used with no data whatsoever.

7 Discussion

Our experiments show that despite its simplicity, **epsinas** metric shows surprisingly good performance across different benchmark datasets. It shows the most consistent performance with NAS-Bench-201, suggesting that this benchmark defines a relatively easy-to-navigate search space. Tables 2, 4, 5 demonstrates that **epsinas** outperforms most of other zero-cost metrics in terms of Spearman and Kendall correlations. The differences are statistically significant, with p-values between **epsinas** and other metrics below 0.001.

Notably, we highlight the key differences between our previous work (cv metric, Gracheva (2021)) and the current approach. The two methods differ in how they de-emphasize weights and the type of outputs used for statistical computation. Using two fixed weight initialisations more effectively isolates the influence of network geometry compared to averaging over a hundred random initialisations. Additionally, the `epsinas` metric is significantly more efficient, reducing computation time by a factor of fifty.

We also compare the performance of `epsinas` with its conceptual counterparts, `zico` and ξ -`gsnr`. While the metrics are based on similar principles, `zico` and ξ -`gsnr` use gradients instead of raw outputs, which appears to limit their performance. This difference may also be attributed to random initialisation, as opposed to the shared constant weight initialisation used by `epsinas`. In all cases, `epsinas` consistently outperforms the naive `nparams` proxy, which relies solely on the total number of parameters in a given network.

It is important to note that we used NAS-Bench-201 to develop our metric, so the findings from this part of the study may be biased. To test the generalizability of the metric, we applied it to the NAS-Bench-101 benchmark. As shown in Table 4, `epsinas` continues to demonstrate strong performance. The peculiar vertical lines in the results are due to clustering based on test accuracy. Since each architecture in NAS-Bench-101 is trained with three random seeds, the four clusters represent cases where training diverges for certain random initialisations. Using more random seeds would likely reveal a smoother pattern and stronger correlations.

By design, we expect our metric to be independent of the specific neural architecture structure. To confirm this claim, we show its applicability to architectures to RNN models with NAS-Bench-NLP.

While the statistics in Table 5 may not seem extraordinary at first glance, Figure 2 demonstrates a visible trend of improved architecture selection as the `epsinas` metric value increases. This positive trend is particularly noteworthy, especially considering the noise inherent to the benchmark. Factors such as the limited sample size of networks within a vast architecture space, variations in training hyperparameters, dropout rates, and other variables can contribute to performance inconsistencies in NAS. Despite these challenges, the trend is strong and consistent enough to confidently conclude that the `epsinas` metric shows great potential for effectively evaluating recurrent-type architectures.

While `epsinas` metric shows solid empirical performance, the underlying reasons for this remain unclear.

There are several hints towards its understanding. First, mathematically, `epsinas` represents the difference in the output distribution shapes between initialisations. The output shape is affected by layer widths, activation functions, batch normalisation, skip connections and other factors, which we generally call network geometry. By setting weights to a constant shared value, one can probe the effects of the geometry without being obstructed by the randomness of initialisation.

Second, according to the weight ablation studies (Section 6), the best performance is achieved when we set the weights to the lowest and highest values that do not lead to excessive output explosion or vanishing. Therefore, `epsinas` measures the amplitude of the outputs' distribution shape change due to geometry.

Finally, we see that grey-scale solid images work reasonably well as inputs during the synthetic data studies. The distribution over the input samples is uniform, which makes it easier to track the changes in the distribution shape as the signal propagates through the network.

7.1 Future directions

While the presented `epsinas` metric demonstrates solid performance, several open questions remain. One of the key directions for future development is to assess its applicability to more types of neural topologies, such as transformer [Chen et al \(2021a\)](#); [Zhou et al \(2024\)](#), generative adversarial [Gao et al \(2020\)](#) or graph neural networks [Ru et al \(2020\)](#), and to test it with hierarchical search spaces ([Christoforidis et al 2021](#); [Ru et al 2020](#); [Liu et al 2019](#); [Chen et al 2021a](#)). It is also interesting to evaluate the metric in combination with other advanced NAS algorithms, such as multi-objective optimization ([Zhao et al 2021](#)), differentiable architecture search [Xiang et al \(2023\)](#), Bayesian optimization ([Snoek et al 2012](#); [Shen et al 2023](#)), or within the Parametric Zero-Cost Proxies ([Dong et al 2024](#)) framework.

The metric’s performance is influenced by various hyperparameters, including weight values, batch size, and initialization scheme. While we conducted ablation studies on their individual effects, potential interactions between these hyperparameters remain unexplored, making it important to investigate their joint impact. Additionally, the current weight setup is based on empirical choices and would benefit from a solid theoretical foundation.

A deeper theoretical study of the metric is particularly needed given its similarities to the `zico` metric. For instance, it remains unclear why the mean absolute error over raw outputs outperforms the deviation over gradients and how this behaviour relates to the initialization scheme. Furthermore, examining the performance of the `zico` metric when initialized with constant shared weights could provide further insights.

8 Conclusion

This work presents a simple zero-cost NAS scoring metric, `epsinas`, which measures how much a network’s outputs change when initialized with two different constant shared weights. We compute it by performing two forward passes and calculating the normalized MAE between the raw outputs.

For a simple metric, `epsinas` shows an unexpectedly high correlation with network performance in CNNs and RNNs, outperforming most existing zero-shot NAS proxies. We evaluate the `epsinas` metric on three widely-used NAS search spaces: NAS-Bench-201, NAS-Bench-101, and NAS-Bench-NLP. Our results show strong performance across these search spaces.

The metric is computationally efficient, requiring only a fraction of a GPU second to evaluate a single architecture, with minimal memory demand, as it does not require gradient computation or labelled data. `epsinas` can be run on a CPU, making it broadly accessible. While small to medium-sized networks can be processed efficiently on a CPU, larger architectures may require GPU acceleration to handle forward passes more quickly, especially when evaluating multiple candidate architectures in

succession. Optimizing CPU-GPU workload distribution could help balance energy consumption and speed in resource-constrained environments such as mobile and embedded systems. In our implementation, we have already incorporated a degree of heterogeneous computing by offloading output data from the GPU to the CPU for statistical computation. Such optimisations could be valuable in domains where rapid evaluation of complex architectures is necessary.

Experimental results demonstrate that `epsinas` applies to convolutional networks and generalizes well to recurrent networks—with careful consideration for embedding layer initialization. The method can also be readily incorporated into other NAS frameworks, such as random search and evolutionary algorithms.

The main downside of our metric is that the optimal weight values need to be set separately for different search spaces and neural topologies. We found that using both high and low weight values helps highlight differences in initialization, leading to better correlations with final performance. While the weights should cover both extremes to test the architecture’s robustness, it’s unclear how to automate this selection process.

These results offer an intriguing clue about the inner workings of neural networks. We hope that `epsinas`’s performance can be improved with a better theoretical understanding of the optimal weight setup. We believe that further development of this method will lead to a faster and computationally efficient NAS and a clearer understanding of network prediction and generalization mechanisms.

These results might offer some interesting clues about how neural networks work. We are hopeful that with a better theoretical understanding of the optimal weight setup, `epsinas`’s performance can improve, and we believe that further development of this method could help us better understand network prediction and generalization.

Declarations

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

Funding

No funding was received for conducting this study.

Code availability

The code supporting this study can be found on GitHub at <https://github.com/egracheva/epsinas>.

References

- Abdelfattah MS, Mehrotra A, Dudziak L, et al (2021) Zero-cost proxies for lightweight nas. arXiv preprint arXiv:210108134

- Agarap AF (2018) Deep learning using rectified linear units (relu). arXiv preprint arXiv:180308375
- Chen B, Li P, Li C, et al (2021a) Glit: Neural architecture search for global and local image transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 12–21
- Chen W, Gong X, Wang Z (2021b) Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. arXiv preprint arXiv:210211535
- Chrzaszcz P, Loshchilov I, Hutter F (2017) A downsampled variant of imagenet as an alternative to the cifar datasets. arXiv preprint arXiv:170708819
- Christoforidis A, Kyriakides G, Margaritis K (2021) A novel evolutionary algorithm for hierarchical neural architecture search. arXiv preprint arXiv:210708484
- Chu X, Wang X, Zhang B, et al (2020) Darts-: robustly stepping out of performance collapse without indicators. arXiv preprint arXiv:200901027
- Dauphin YN, Bengio Y (2013) Big neural networks waste capacity. arXiv preprint arXiv:13013583
- Deng B, Yan J, Lin D (2017) Peephole: Predicting network performance before training. arXiv preprint arXiv:171203351
- Dong P, Li L, Pan X, et al (2024) Parzc: Parametric zero-cost proxies for efficient nas. arXiv preprint arXiv:240202105
- Dong X, Yang Y (2019) Searching for a robust neural architecture in four gpu hours. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 1761–1770
- Dong X, Yang Y (2020) Nas-bench-201: Extending the scope of reproducible neural architecture search. arXiv preprint arXiv:200100326
- Falkner S, Klein A, Hutter F (2018) Bohb: Robust and efficient hyperparameter optimization at scale. arXiv preprint arXiv:180701774
- Fan Y, Niu ZH, Yang YB (2023) Data-aware zero-shot neural architecture search for image recognition. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 1–5
- Frankle J, Carbin M (2018) The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:180303635
- Gaier A, Ha D (2019) Weight agnostic neural networks. Advances in neural information processing systems 32

- Gao C, Chen Y, Liu S, et al (2020) Adversarialnas: Adversarial neural architecture search for gans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
- Gracheva E (2021) Trainless model performance estimation based on random weights initialisations for neural architecture search. *Array* 12:100082. doi:<https://doi.org/10.1016/j.array.2021.100082>
- Istrate R, Scheidegger F, Mariani G, et al (2019) Tapas: Train-less accuracy predictor for architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp 3927–3934
- Klyuchnikov N, Trofimov I, Artemova E, et al (2022) Nas-bench-nlp: neural architecture search benchmark for natural language processing. *IEEE Access* 10:45736–45747
- Krizhevsky A, Hinton G, et al (2009) Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto
- LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Advances in neural information processing systems, pp 598–605
- LeCun Y, Cortes C, Burges C (2010) Mnist handwritten digit database. ATT Labs [Online] Available: <http://yannlecun.com/exdb/mnist> 2
- Lee N, Ajanthan T, Torr PH (2018) Snip: Single-shot network pruning based on connection sensitivity. arXiv preprint arXiv:181002340
- Li G, Yang Y, Bhardwaj K, et al (2023) Zico: Zero-shot nas via inverse coefficient of variation on gradients. arXiv preprint arXiv:230111300
- Li G, Hoang D, Bhardwaj K, et al (2024) Zero-shot neural architecture search: Challenges, solutions, and opportunities. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46(12):7618–7635. doi:[10.1109/TPAMI.2024.3395423](https://doi.org/10.1109/TPAMI.2024.3395423)
- Li L, Talwalkar A (2020) Random search and reproducibility for neural architecture search. In: Uncertainty in artificial intelligence, PMLR, pp 367–377
- Lin M, Wang P, Sun Z, et al (2021) Zen-nas: A zero-shot nas for high-performance image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 347–356
- Liu C, Chen LC, Schroff F, et al (2019) Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
- Liu H, Simonyan K, Yang Y (2018) Darts: Differentiable architecture search. arXiv preprint arXiv:180609055

- Liu Y, Sun Y, Xue B, et al (2021) A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems* 34(2):550–570
- Lopes V, Santos M, Degardin B, et al (2022) Efficient guided evolution for neural architecture search. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp 655–658
- Marcinkiewicz MA (1994) Building a large annotated corpus of english: The penn treebank. Using Large Corpora 273
- Mellor J, Turner J, Storkey A, et al (2020) Neural architecture search without training. arXiv preprint arXiv:200604647v1
- Ning X, Tang C, Li W, et al (2021) Evaluating efficient performance estimators of neural architectures. *Advances in Neural Information Processing Systems* 34:12265–12277
- Northcutt CG, Athalye A, Mueller J (2021) Pervasive label errors in test sets destabilize machine learning benchmarks. arXiv preprint arXiv:210314749
- Pham H, Guan MY, Zoph B, et al (2018) Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:180203268
- Real E, Aggarwal A, Huang Y, et al (2019) Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence, pp 4780–4789
- Ru R, Esperanca P, Carlucci FM (2020) Neural architecture generator optimization. *Advances in Neural Information Processing Systems* 33:12057–12069
- Salmani Pour Avval S, Eskue ND, Groves RM, et al (2025) Systematic review on neural architecture search. *Artificial Intelligence Review* 58(3):73
- Sandler M, Howard A, Zhu M, et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
- Shen Y, Li Y, Zheng J, et al (2023) Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies. In: Proceedings of the AAAI conference on artificial intelligence, pp 9792–9801
- Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* 25
- Sun Z, Sun Y, Yang L, et al (2023) Unleashing the power of gradient signal-to-noise ratio for zero-shot nas. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 5763–5773

- Tanaka H, Kunin D, Yamins DL, et al (2020) Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems* 33:6377–6389
- Wang C, Zhang G, Grosse R (2020) Picking winning tickets before training by preserving gradient flow. *arXiv preprint arXiv:200207376*
- White C, Neiswanger W, Savani Y (2021) Bananas: Bayesian optimization with neural architectures for neural architecture search. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp 10293–10301
- White C, Khodak M, Tu R, et al (2022) A deeper look at zero-cost proxies for lightweight nas. *ICLR Blog Track*
- Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3):229–256
- Xiang L, Dudziak L, Abdelfattah MS, et al (2023) Zero-cost operation scoring in differentiable architecture search. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, pp 10453–10463
- Ying C, Klein A, Christiansen E, et al (2019) Nas-bench-101: Towards reproducible neural architecture search. In: *International Conference on Machine Learning*, PMLR, pp 7105–7114
- Zhang Z, Jia Z (2021) Gradsign: Model performance inference with theoretical insights. *arXiv preprint arXiv:211008616*
- Zhao Y, Wang L, Yang K, et al (2021) Multi-objective optimization by learning space partitions. *arXiv preprint arXiv:211003173*
- Zhou Q, Sheng K, Zheng X, et al (2024) Training-free transformer architecture search with zero-cost proxy guided evolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*

A Visualisation of other zero-cost NAS metrics

A.1 When integrated with other NAS methods

In Section 5.6, we show how `epsinas` metric improves the performance of ageing evolution and random search when used for warming up. Figure 7 compares `epsinas` integration to other metrics from [Abdelfattah et al \(2021\)](#). For both EA and RS, we use the metrics for a warm-up and run the procedure until the number of trained architectures reaches 300 with 100 random rounds. The warm-up pool contains 3000 randomly selected architectures.

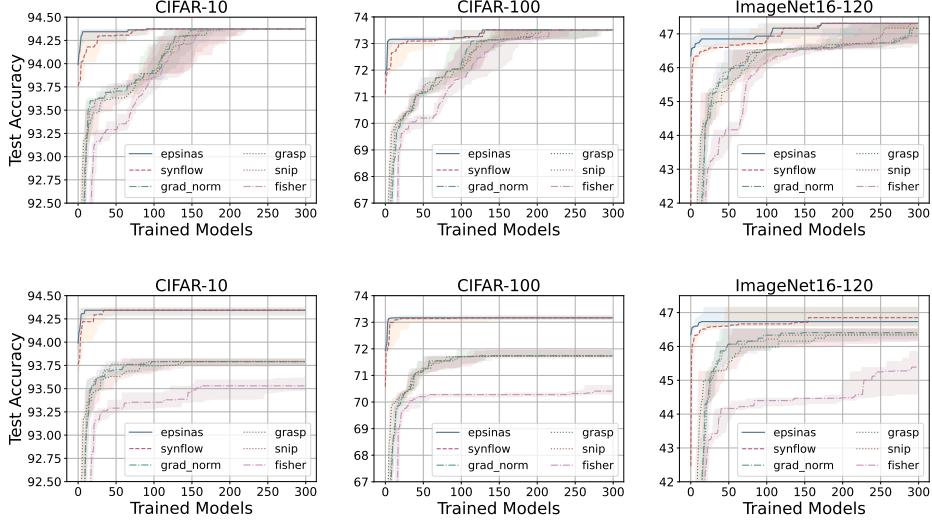


Fig. 7 Various zero-cost NAS metrics’ performance when used as a warm-up for ageing evolution (top) and random search (bottom) for three datasets from the NAS-Bench-201 search space. Shadows span between lower and upper quartiles based on 100 rounds. The warm-up population is 3000 architectures.

A.2 Correlation with accuracy

In [Abdelfattah et al \(2021\)](#), the metrics are presented through statistical measures, but we feel that visualisation helps to improve understanding. Here, we provide visualisations for two search spaces built on the data provided by the authors.

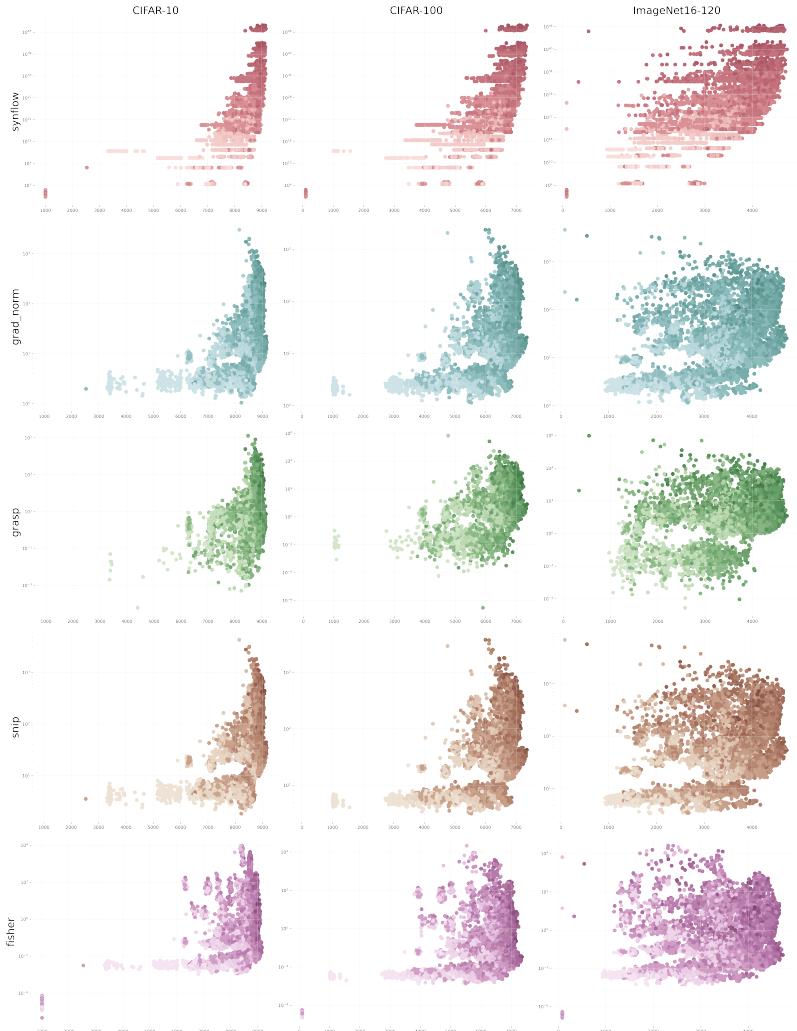


Fig. 8 Zero-cost metrics performance illustration for NAS-Bench-201 search space evaluated on CIFAR-10, CIFAR-100 and ImageNet16-120 datasets, based on data from [Abdelfattah et al \(2021\)](#). Each dot corresponds to an architecture; the darker the colour, the more parameters it contains. The figure represents the search space of 15,625 networks (excluding architectures with NaN scores).